Embedded Linux Conference
Europe

Device Tree: Past, Present, Future

Neil Armstrong - BayLibre

THE LINUX FOUNDATION

Quick bio

Embedded Linux Engineer since *2008*

First Embedded Linux Conference Europe in 2009

Linux Device Tree adopter since *2012*

BayLibre Kernel Hacker since *2015*

Linux & U-Boot Maintainer since *2017*

Device Tree: Past

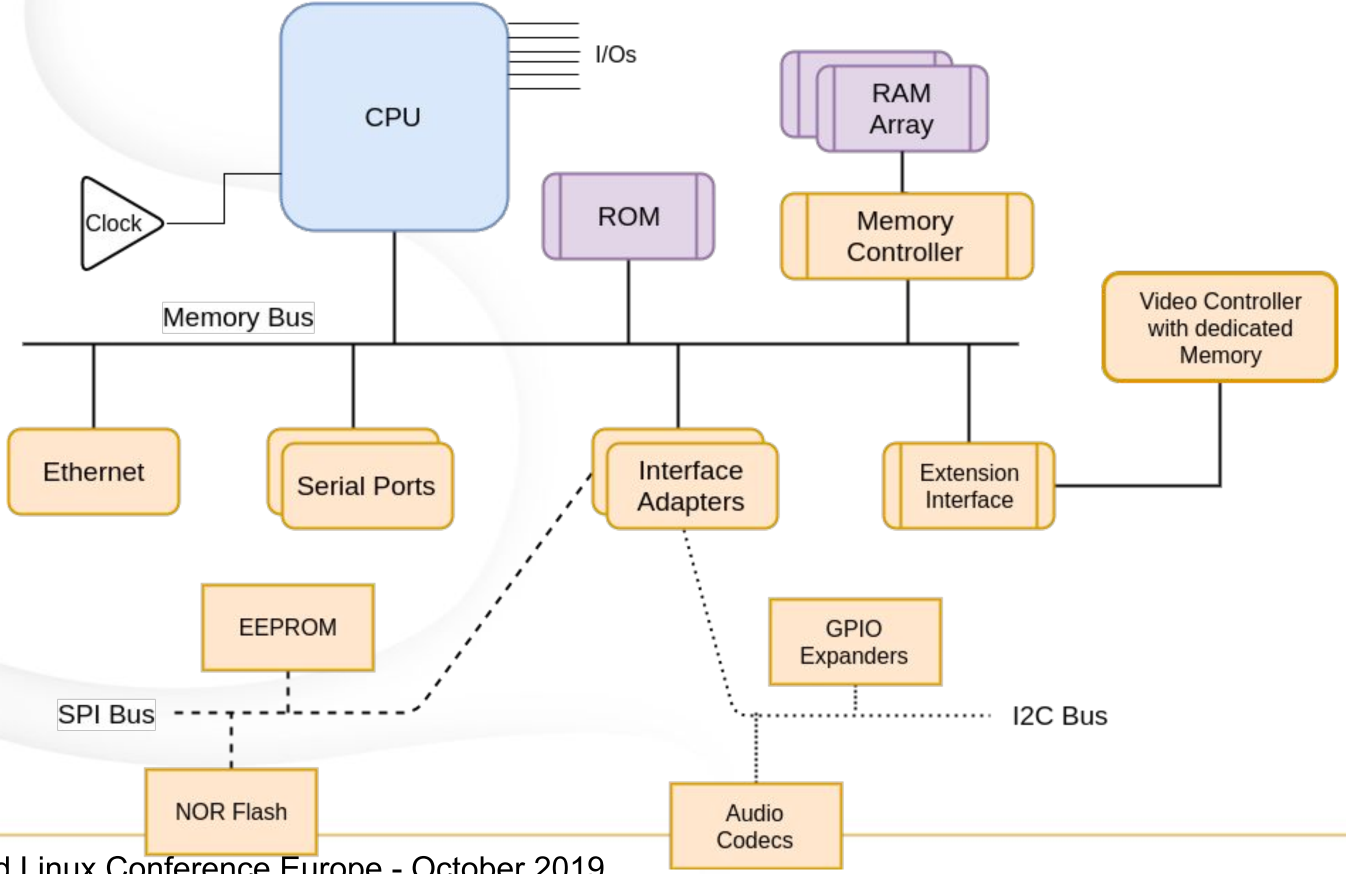
Where does it comes from ?

Device Tree: Past

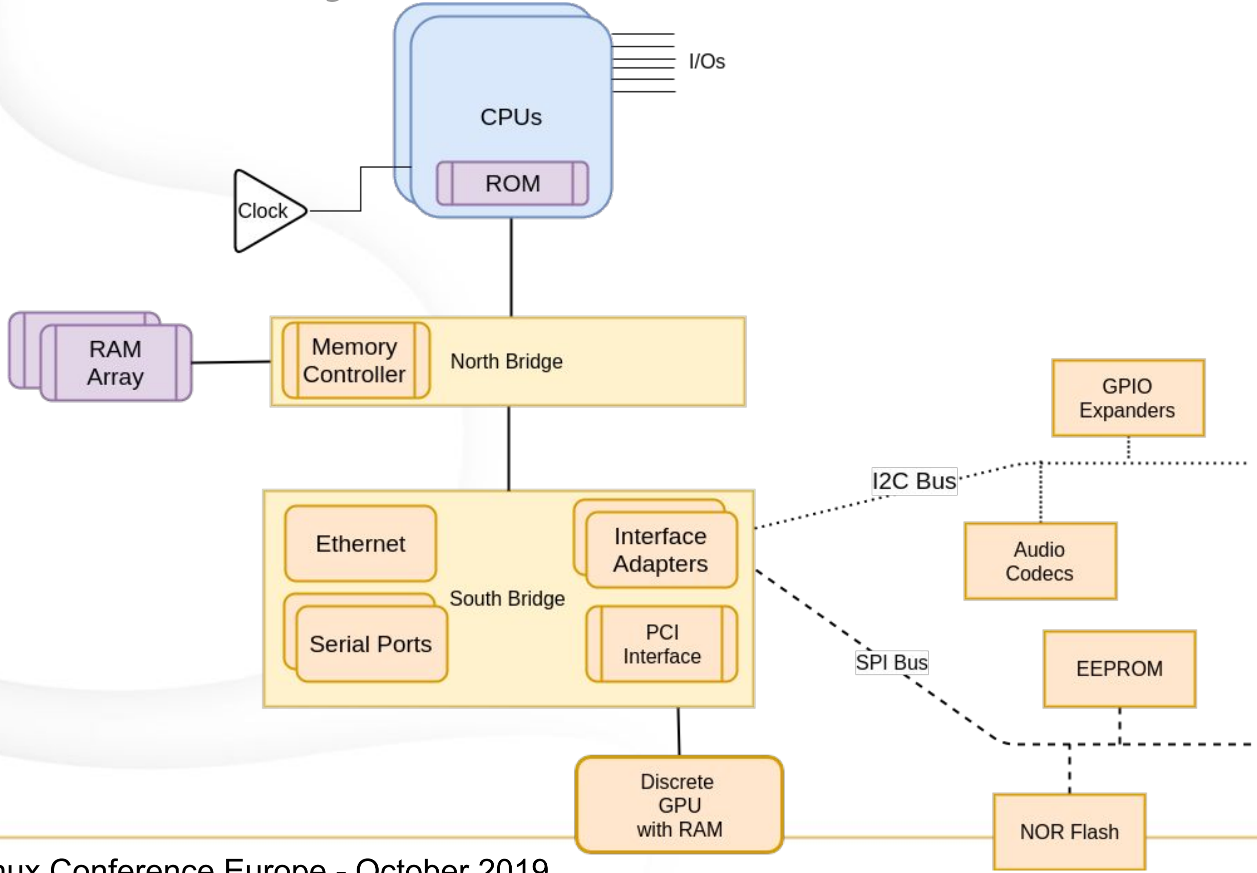
Software Engineers always struggled to describe in a simple and portable way the different hardwares.

- In code ?
- In binary format ?
- In text format ?
- Auto generated ?

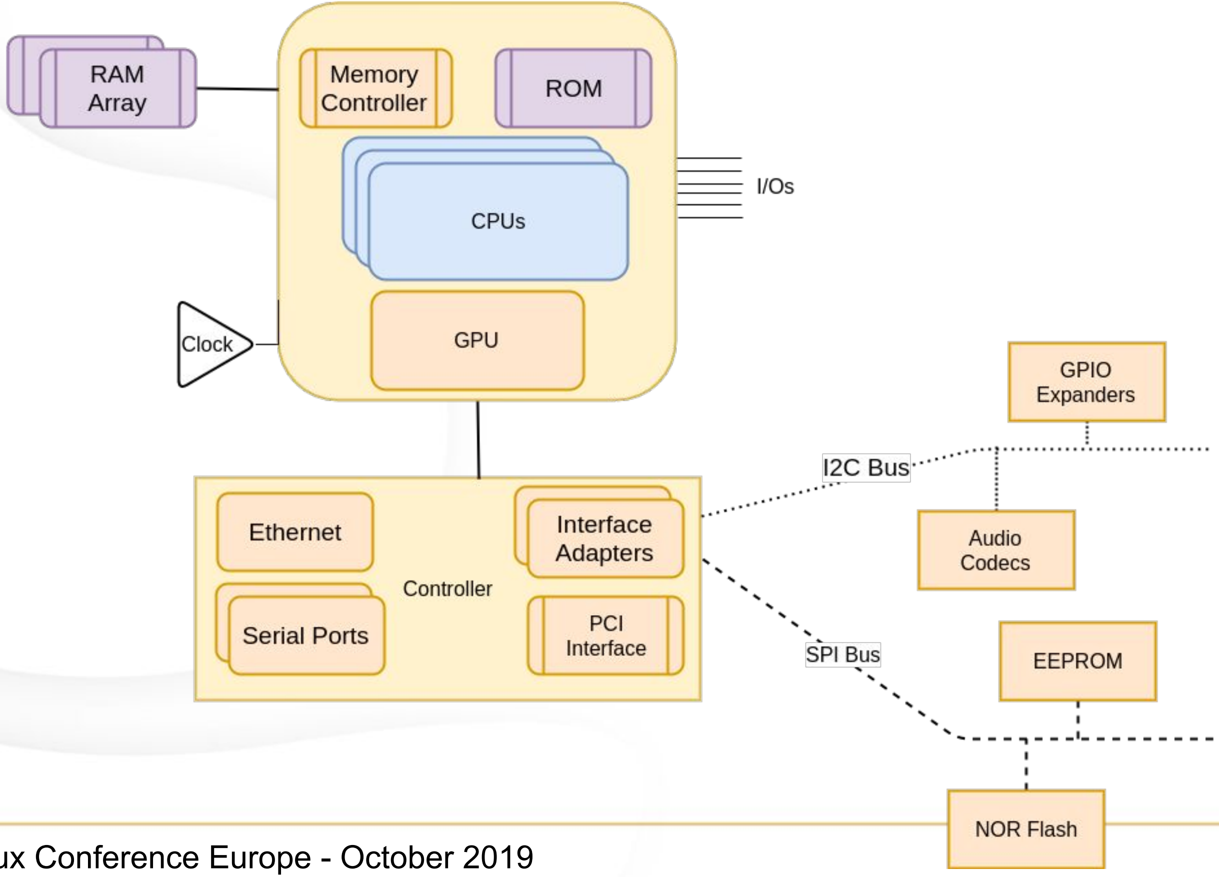
Classic System Architecture



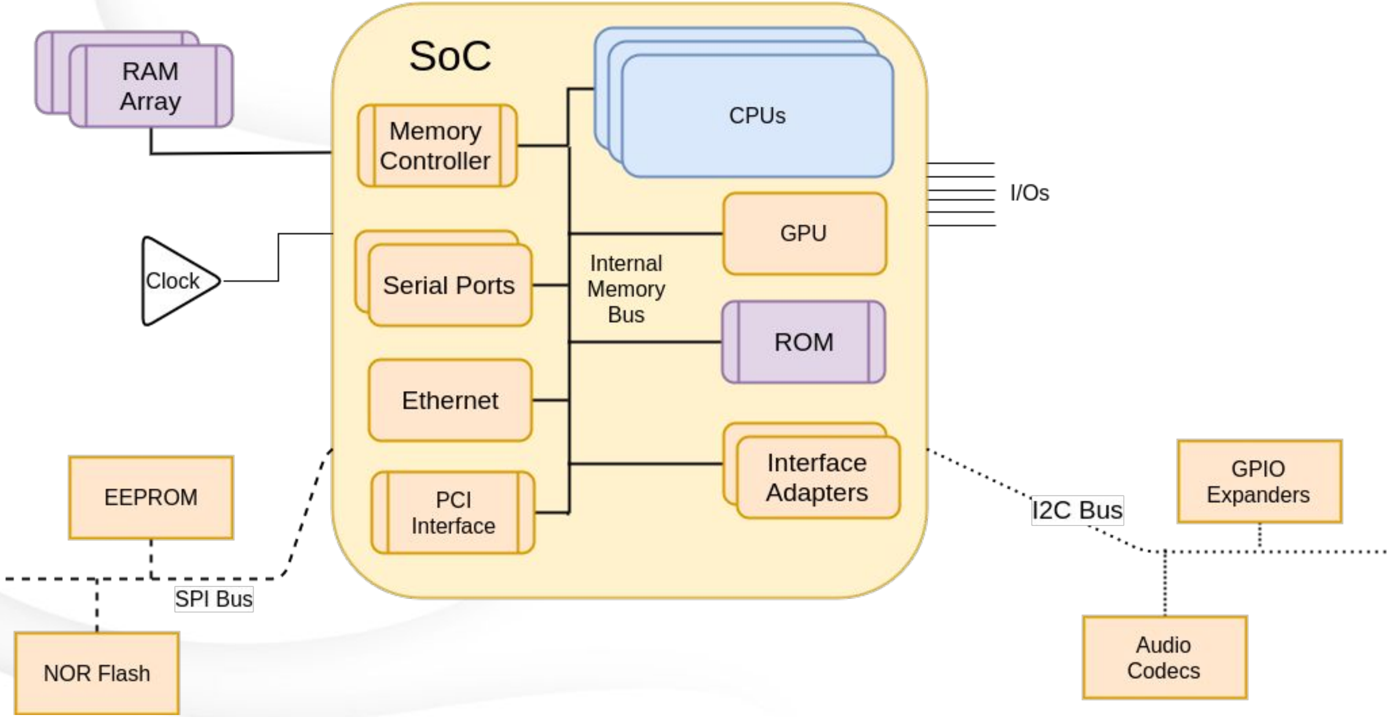
Classic x86 System Architecture



Modern System Architecture



System-On-Chip Architecture



Device Tree : Past

- Multiple different solutions were found :
  - ACPI table in the x86 world
  - Device Tree in the SPARC/PowerPC world
- Both solutions describes the hardware in a generic way that can be parsed and understood by the Operating System
- ACPI was designed to be very Intel/Microsoft specific, even if it was used on various architectures with Windows NT

Device Tree : Past

- ACPI is a very complex hardware description, aimed to keep a tight control on the PC architecture & market
  - “Hardware Enumeration” in DSDT ACPI table
  - More than description, defines how to configure and manage HW power states
- Device Tree was designed to be written/understood by humans then processed to be understood by software

Device Tree : Specifications

- IEEE-1275 formalized specification (1994)
  - The **set of devices attached to the system**, including permanently installed devices and plug-in devices, is described by an Open Firmware **data structure known as the device tree**. The operating system may inspect the device tree to determine the hardware configuration of the system. Each **device in the device tree is described by a property list**. The set of properties describing a device is arbitrarily extensible so that any type of device and any kind of information that needs to be reported about the device can be accommodated.

Device Tree : Specifications

- PowerPC™ Common Hardware Reference Platform (CHRP) specifies DT (1995)
  - Developed by Apple, IBM and Motorola
  - Extends IEEE-1275 to PowerPC™
- ePAPR specifies DT (2008)
  - Stands for Power.org™ Standard for Embedded Power Architecture™ Platform Requirements
  - “The ePAPR is loosely related to the IEEE 1275”
  - “Because of the nature of embedded systems, some of these problems faced by open, general purpose computers do not apply”

Device Tree : History

Multiple implementation existed with different formats but the same goal :

- Sun Microsystems - Open Boot / Open Firmware (1988)
  - Used in SPARC systems
  - Uses Device Tree to describe hardware
- Apple adopts Open Firmware on Power Mac 7200 (1995) using DT along IBM
  - Uses Open Firmware and DT until Macs switched to Intel CPUs
  - Still uses a DT form in iPhone/iPad/iPod ARM Based devices
  - PCI devices represented in DT, generated by firmware

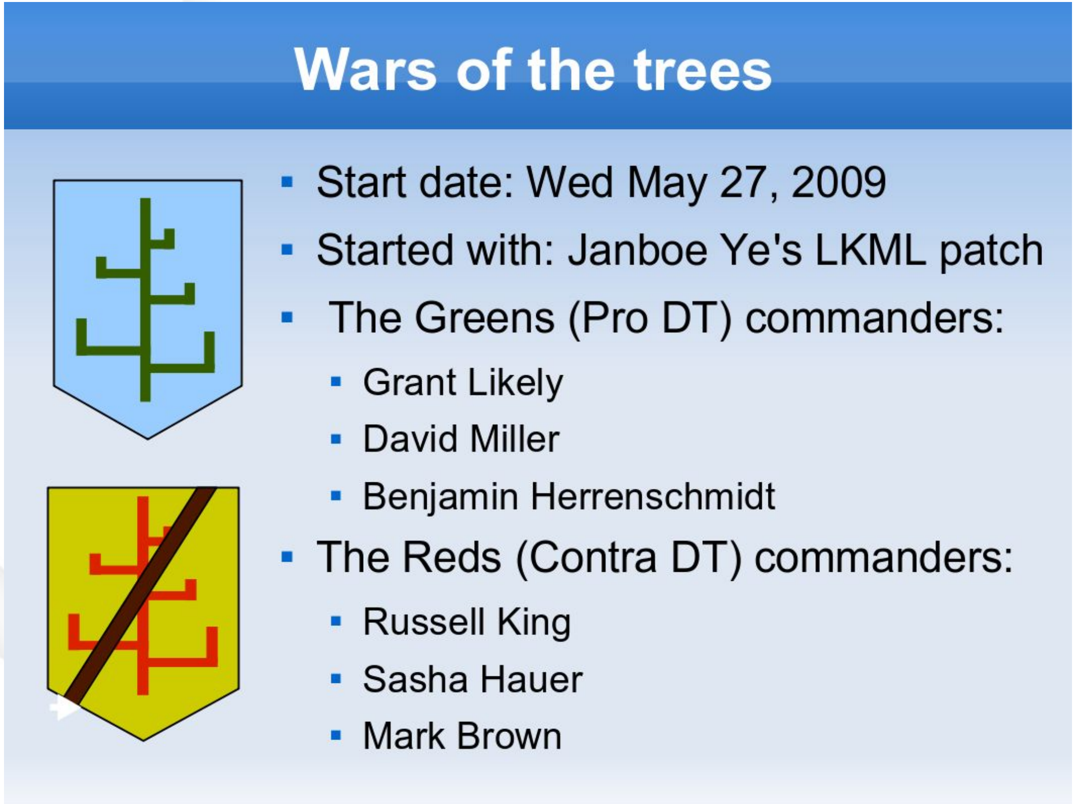
Device Tree : History

- 2005 : Linux PowerPC support starts switching to Device tree
- 2006 : First device tree in Linux codebase “mpc8641_hpcn.dts”
- 2007 :
  - Initial Sony PS3 support with basic DTS
  - Common Device Tree implementation for Sparc and PowerPC in “drivers/of”
- 2009 : microblaze switched to Device Tree
- 2011 : ARM switched to Device Tree
- 2014 : Device Tree used by 12 over 28 architecture

A little bit of ELC history

- Almost 10 years go (October 15 & 16, 2009), Embedded Linux Conference Europe was located in Grenoble (**91,65 km** from here, *56.94 miles*)
- Device Tree for ARM was heavily discussed in 2009 & 2010
- Vitaly Wool and Wolfram Sang did great talks on the topic in 2009
  - **https://elinux.org/ELC_Europe_2009_Presentations**
  - Wolfram shared his experience on I2C, UIO and GPIO for Device Tree at the time
  - Vitaly summed-up the heated discussions about Device Tree
    Spoiler Alert: the “Pro DT” did win the debate

A little bit of ELC history

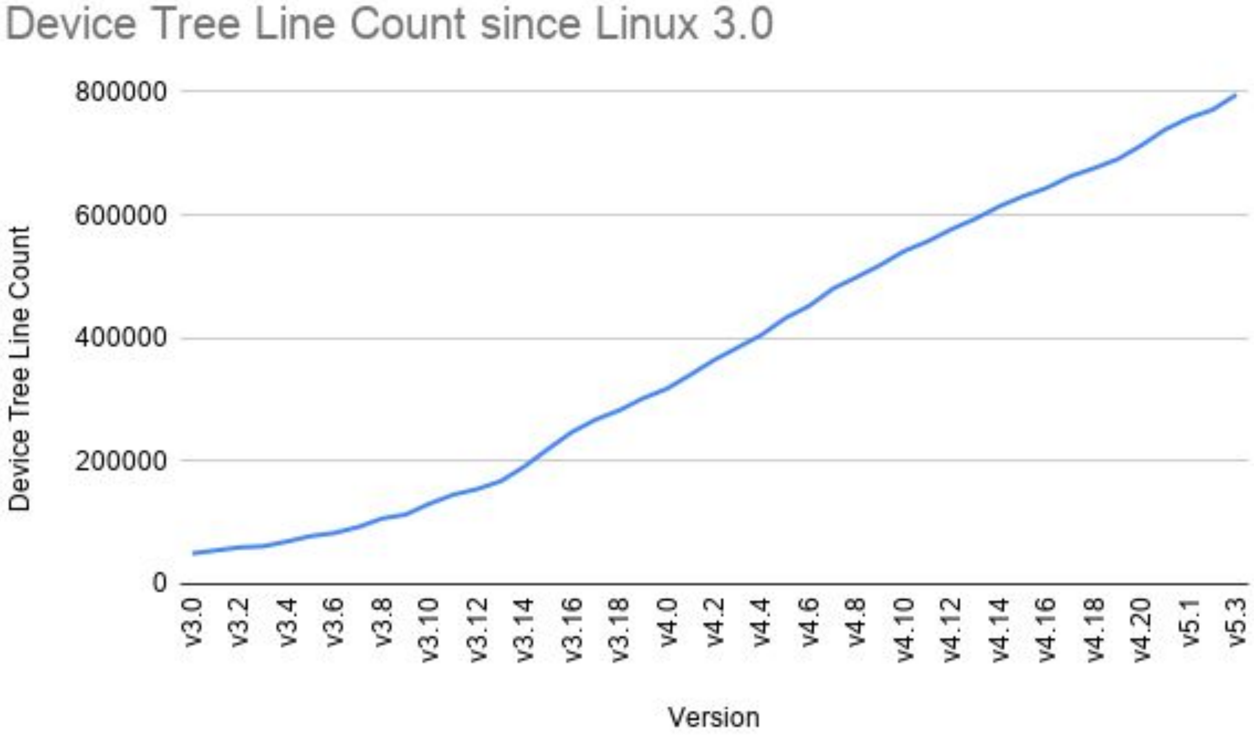


Wars of the trees

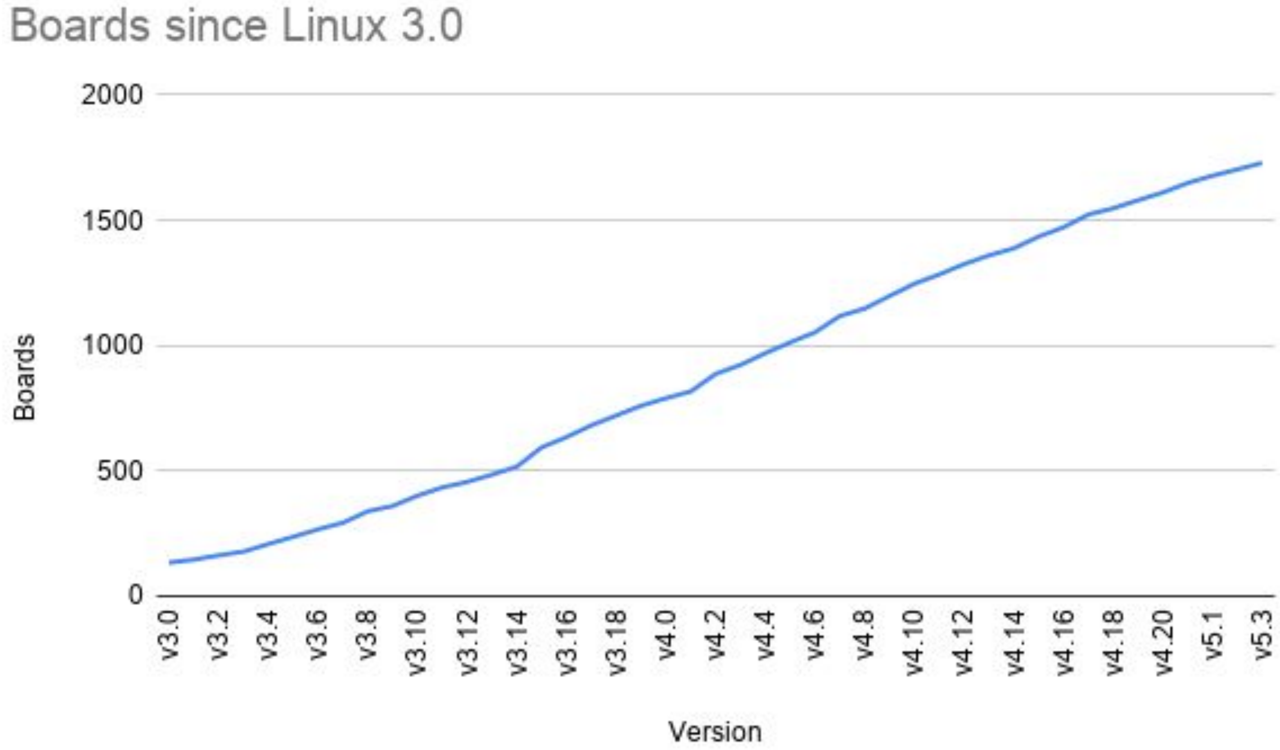
- Start date: Wed May 27, 2009
- Started with: Janboe Ye's LKML patch
- The Greens (Pro DT) commanders:
  - Grant Likely
  - David Miller
  - Benjamin Herrenschmidt
- The Reds (Contra DT) commanders:
  - Russell King
  - Sasha Hauer
  - Mark Brown

Thanks Vitaly for providing the .odp of the slides, still perfectly opening 10y after !

Device Tree : History

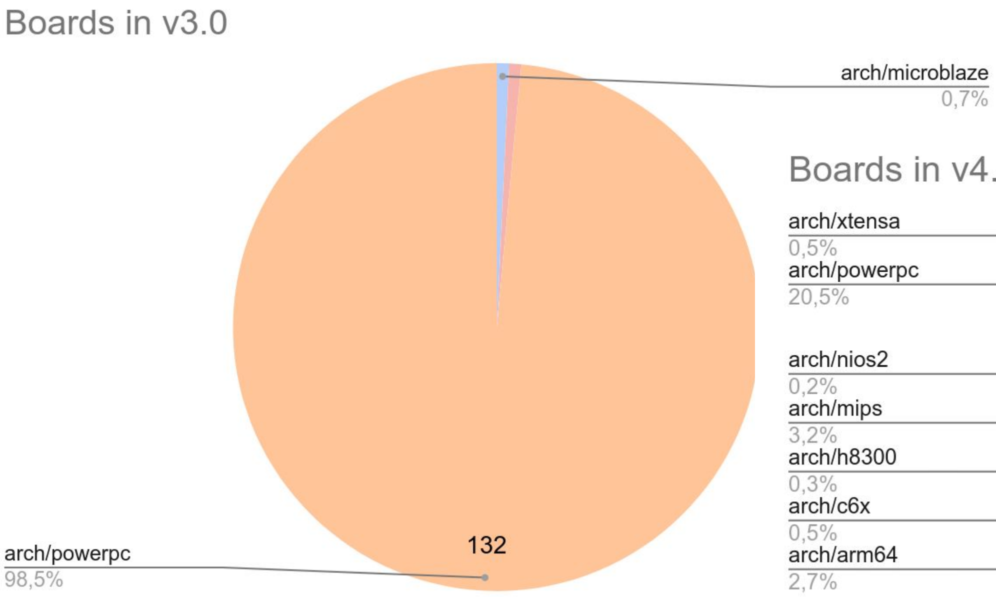


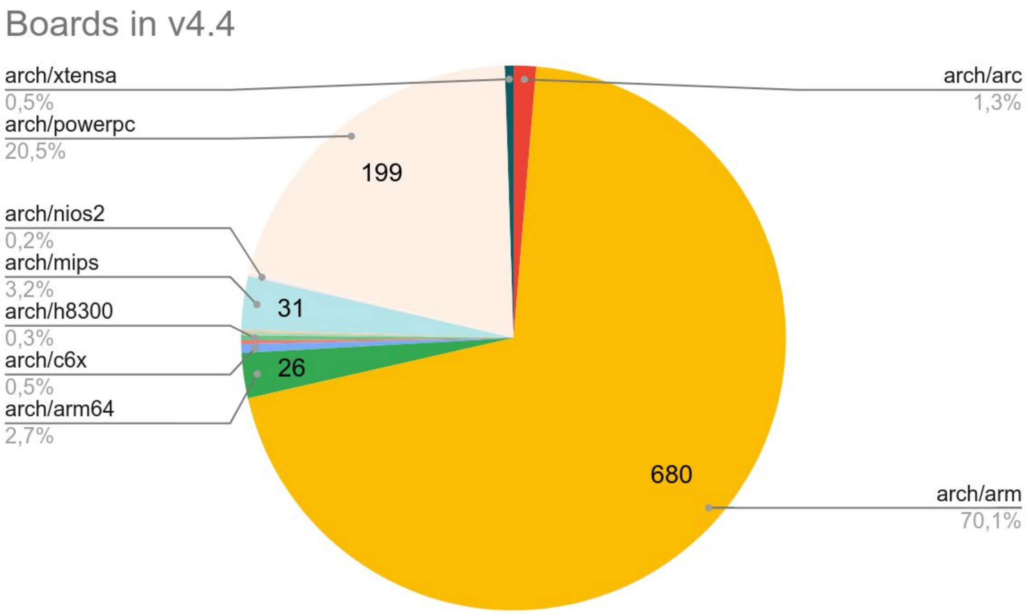
Device Tree : History



Device Tree : History

Embedded Linux Conference Europe

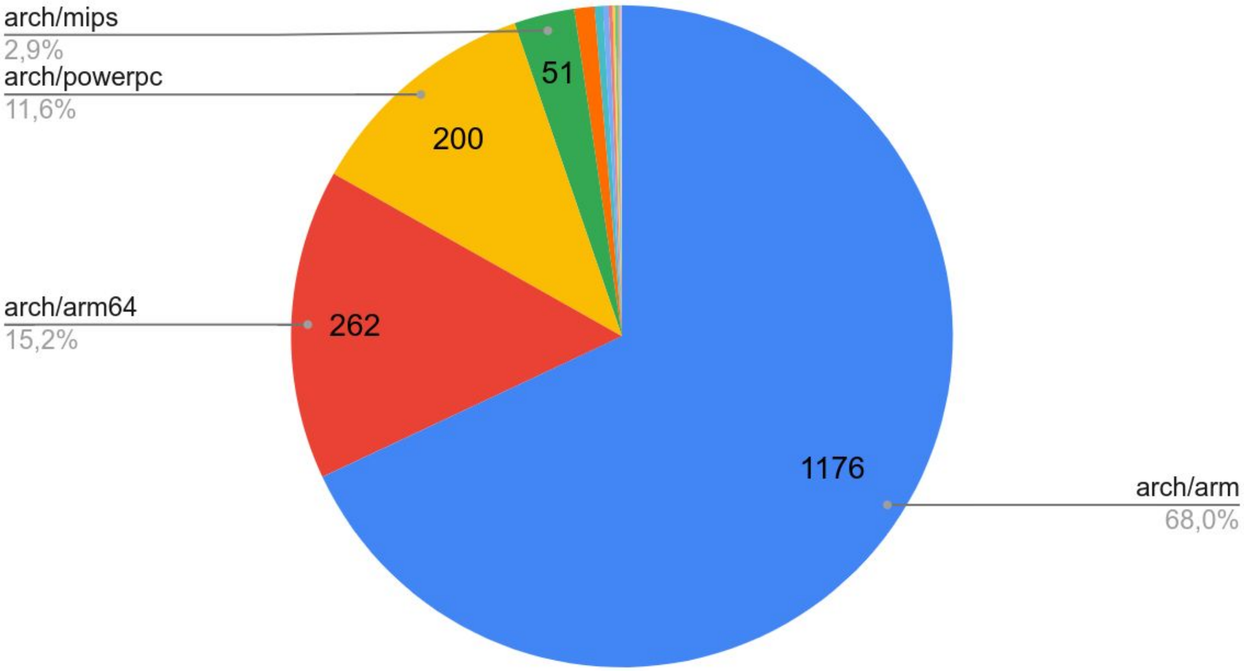




Device Tree : History

Arch	Boards
arch/arm	1176
arch/arm64	262
arch/powerpc	200
arch/mips	51
arch/arc	17
arch/xtensa	7
arch/c6x	5
arch/h8300	3
arch/nios2	2
arch/openrisc	2
arch/microblaze	1
arch/nds32	1
arch/riscv	1
arch/sh	1

Boards in v5.4



Device Tree: Present

How and where is it used ?

Device Tree : Specifications

http://devicetree.org

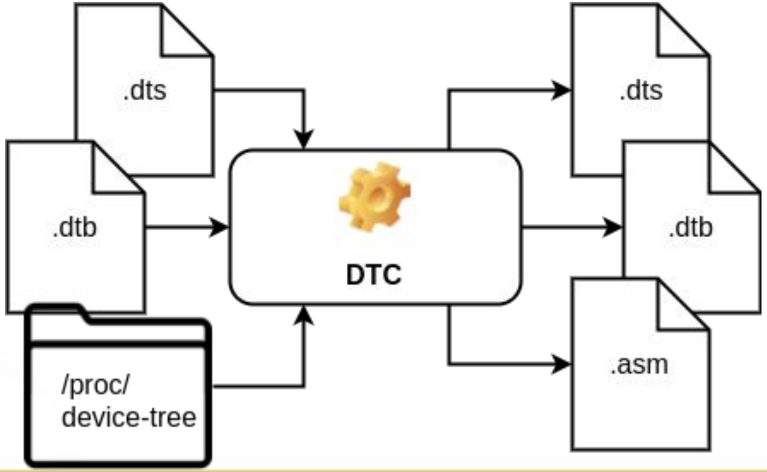
- A new initiative appeared to facilitate the future evolution of the Device Tree Standard.
  - Established by ARM, IBM and NXP
  - Sponsored by ARM, Hisilicon, ST and Linaro
  - Continuation of the ePAPR specifications
  - Adapted to current and future hardware
- v0.2 has been tagged on 20th December 2017
- Community Work is done around a github and a mailing-list

Device Tree: Basics

- Will cover only basics
- You can find extended Device Tree informations in the following :
  - “Engaging Device Trees” Geert Uytterhoeven
  - “Device Tree for Dummies” Thomas Petazzoni
  - “Contemporary Device Tree” Matt Porter
  - “Device Tree The Disaster so Far” Mark Rutland
  - NXP AN5125 “Introduction to Device Trees”
  - https://www.devicetree.org/specifications/
  - Power.org™ Standard for Embedded Power Architecture™ Platform Requirements (ePAPR)

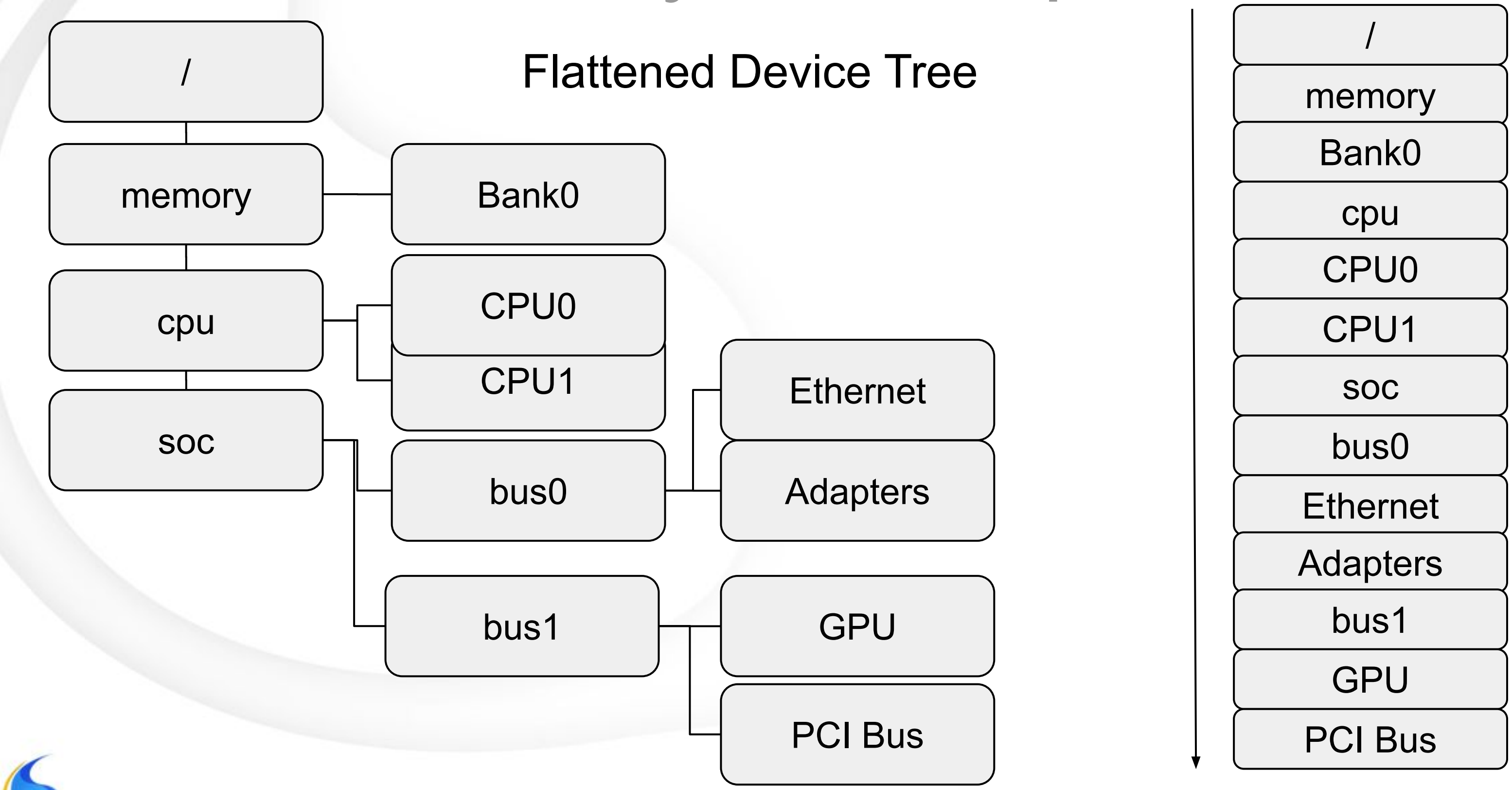
Device Tree: Source Format

- You will find up to **828334** lines of Device Tree source in current Linux Tree
- “**Device Tree Compiler**” aka DTC can “compile” the source format into :
  - Source Format
  - Device Tree Blob : What will be used by U-Boot/Linux
  - Blob in Assembly
- DTC can also read any of these binary formats to source format



Device Tree: System Representation

Flattened Device Tree



Device Tree: Tools

- DTC
  - Official Compiler
  - Maintained by David Gibson & Jon Loeliger
- PyFDT : https://github.com/superna9999/pyfdt
  - My Python implementation of Device Tree data structure
  - Loads Binary Blobs into native object model
  - Can produce Source, Blobs and experimental Json format
- Other ?
  - Zephyr has a custom Python based Source Interpreter
  - DT Schemas tools converts DTB into Yaml for validation

Device Tree: Source Format

- Nodes
  - Groupings of properties and child nodes
- Properties
  - Key-Value Pairs
    - Text strings “my string”
    - Cells (32-bit) <0xdeadbeef 11 0xf00d>
    - Binary data [0x01 0x02 0x03 0x04]
    - mixed data, concatenate with a comma
      - “my string”, <0xdeadbeef>, [0x04], “your string”
- Phandles
  - Reference to another node

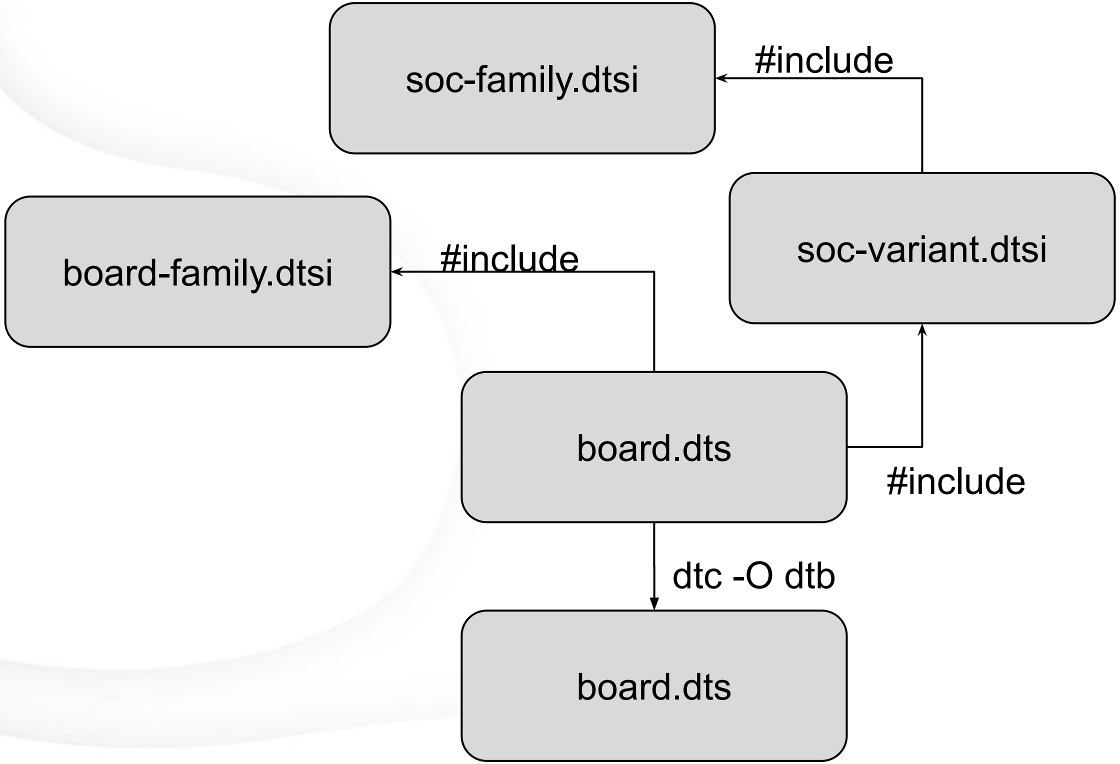
Device Tree: Source Format

```
/ {
    node1@0 {
        a-cell-property = <1 2 3 4>;
        a-string-list-property = "first string", "second string";
        a-byte-data-property = [0x01 0x23 0x34 0x56];
        child-node1@0 {
                first-child-property;
                second-child-property = <1>;
                third-child-property = <&node2 12 235>;
                a-string-property = "Hello, world";
        };
        node2: child-node2@1 {
        };
    };
};
```

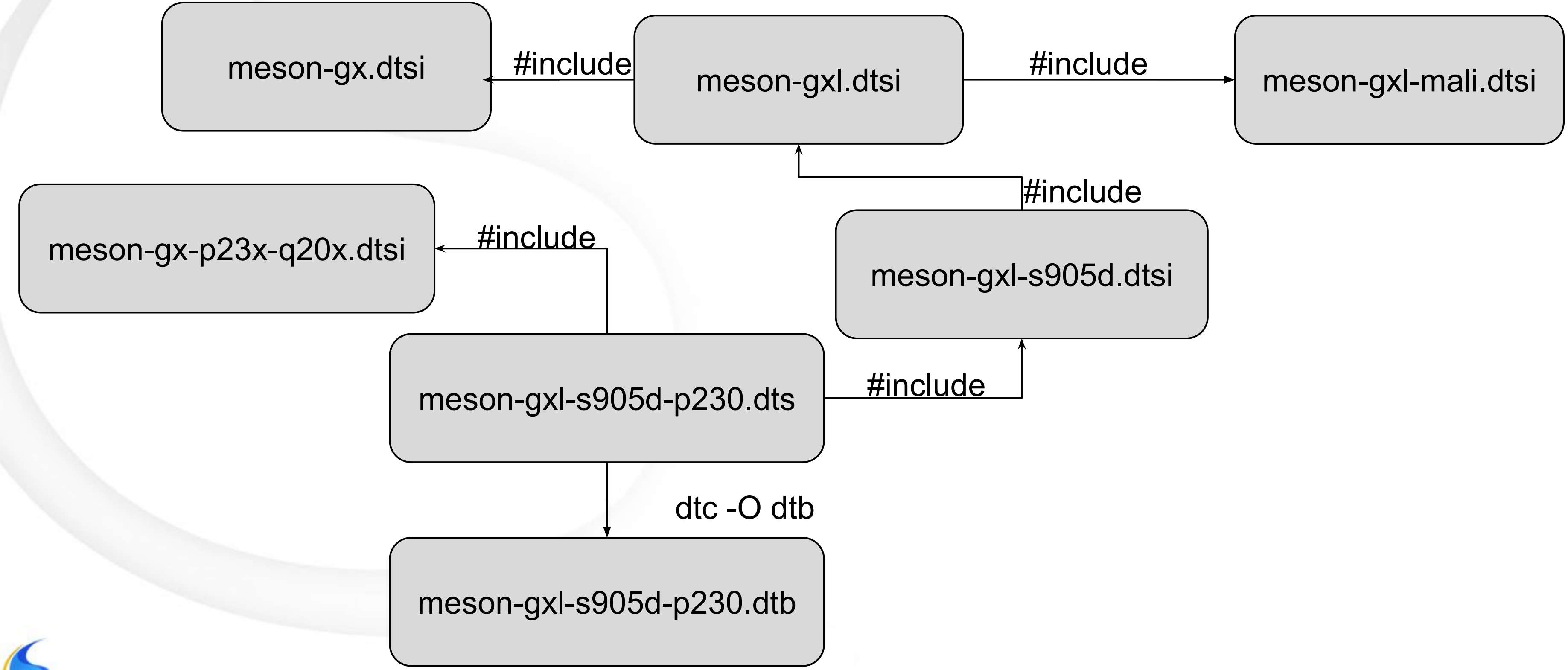
each number (cell) is a uint32

Phandle with parameters

Device Tree: System Representation

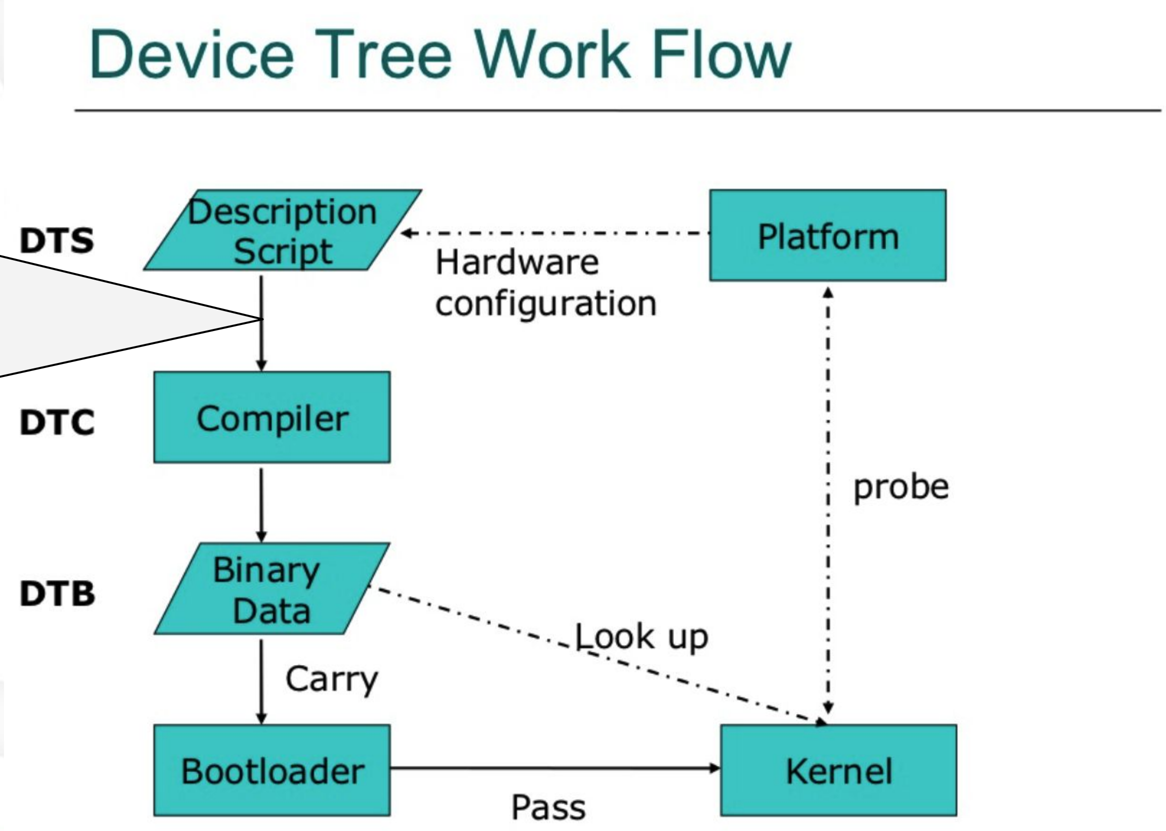


Device Tree: System Representation



Device Tree : Work Flow

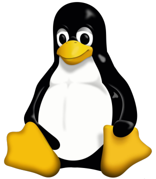
Now, GCC is used to pre-process the DTS Source, to include shared C headers



Device Tree: Present

- Linux
  - Supported in 15 architectures : arc, arm, arm64, c6x, cris, h8300, metag, microblaze, mips, nios2, openrisc, powerpc, **riscv**, sh, xtensa (+ 2 x86 boards)
  - Mandatory for new Hardware in (at least) ARM/ARM64
  - Adopted by **riscv**, DT was in ROM until early this year
  - ARM : 1176 board, ARM64 : 262, powerpc : 200
    - 1438 ARM board (partially) supported ! *(v5.4-rc4)*

Device Tree: Present



- Bindings (**Device-tree schemas LWN.net article)**
  - The device-tree **bindings define how** a particular piece of **hardware is described** in a device tree.
  - **Drivers** then **implement** those bindings.
  - The device-tree **documentation shows how to use the bindings** to describe systems: which properties are available and which values they may have.
  - In theory, the bindings, **drivers and documentation should be consistent** [...] In practice, they are often not consistent and, [...] using those bindings correctly [...] is not a trivial task

Device Tree: Present

- Bindings as Device-tree schemas
  - Recently, Rob Herring proposed a move to a more structured documentation format for device-tree bindings using JSON Schema to allow automated validation.
  - Numerous proposals have been made in the past to address the validation of device trees.
  - Instead of converting DT to YAML, let’s formalize the Bindings
    - Free-form text to YAML
    - JSON Schema for schema vocabulary
    - Permits validating Bindings and DT at build-time

Device Tree: Present

- Bindings as Device-tree schemas

```
$id: "http://devicetree.org/schemas/bindings/vendor/someexample.yaml#"
    $schema: "http://devicetree.org/meta-schemas/core.yaml#"

    title: Documentation example

    maintainers:
        - Our Maintainer <example@example.com>

    description:  |
        Multi-line description is to be added here.

    properties:
        # Here we define the compatible property with one possible string
        compatible:
            items:
                - const: vendor,my-clk
        reg:
            maxItems: 1

    required:
        - compatible
```

Device Tree: Present

- BSD*
  - FreeBSD : support for ARM, AVR32, MIPS, PowerPC based platforms
  - NetBSD: “Flat device tree” support was added in version 8.0 for NetBSD/evbarm (aarch32 and aarch64)
  - OpenBSD: supported by sync’ing Linux DTBs

Device Tree: Present

- U-Boot
  - Used in main U-Boot with the Linux Device Trees along the U-Boot “Driver Model”
  - Used as Boot Image in Flattened Image Tree format to store multiple kernels and add security
  - Supports loading Overlays before starting the kernel
- U-Boot Device Tree support in SPL
  - SPL (Secondary Program Loader) is memory constraint
  - Nodes to be kept at tagged with “u-boot,dm-spl”

Device Tree: Present

- EFI
  - Can pass a Device Tree for non-ACPI ARM/ARM64/RISCV systems
- Trusted Firmware-A
  - Since 2013, first Foundation Model, now STM32MP1
- OPTEE
  - Since Nov 30, 2018 for STM32MP1 support

Device Tree: Non-Linux

- Used Zephyr RTOS
  - The DeviceTree is not used at runtime, but used to generate some compile time defines used by:
    - the build system (CMake)
    - the config system (Kconfig)
  - This gives more flexibility over C header files or Kconfig config files
  - Can leverage the concept of Overlays
  - Shares the same memory constraint as U-Boot SPL

Device Tree: Future

Where and how will it be used ?

Device Tree: Future

- Linux Plumbers **Devicetree track** *November 2018*
    - https://elinux.org/Device_tree_future#Linux_Plumbers_2018
    - Json-schema for DT bindings and validation
    - DT memory (kernel), DT memory (bootloader), storage (FDT) size
    - New FDT format & Overlays
    - FPGA and Devicetree

Device Tree: Future

- Overlays (even if it was already been partially mainlined)
  - Overlays permits dynamically changing the Device Tree on a running system
  - Is used a lot on Single Board Computers
  - Very handy to support HATs or Daughter Boards
  - Plumbing is already in the kernel for years
  - But still in discussion about how to handle maintenance of these overlays fragments.

Device Tree: Future

- Current Overlays as used out-of-tree (e.g. RPi)
  - You can specify which node you which to change
  - You can specify multiple changes : fragments
  - Syntax Example :

~~fragment@0 {~~

~~tar~~

~~node1: subnode {~~

~~ny-attribute = “abcd”;~~

~~};~~

~~};~~

Deprecated syntax, don’t use anymore !

Device Tree: Future

- Overlays
  - Not yet totally formalized
  - Official DTC compiler supports syntax since October 2018
  - Bootloader (U-Boot) can load overlays before loading Linux
  - A lot of issues to fix/solve for live Linux Overlay loading
    - Overlay validation
    - Metadata encoding
    - New format of overlay is in discussion

Embedded Linux Conference
Europe

Any questions ?

Thank you !